```
module fifo_sync #(
    parameter DATA_WIDTH = 8,     // Width of each data word
    parameter FIFO_DEPTH = 16     // Depth (number of words)
)(
    input wire clk,
    input wire rst,

    input wire wr_en,             // Write enable
    input wire rd_en,             // Read enable

    input wire [DATA_WIDTH-1:0] din,   // Data input
    output reg [DATA_WIDTH-1:0] dout,  // Data output

    output wire full,
    output wire empty,
    output wire almost_full,
    output wire almost_empty
);

    // Memory array
    reg [DATA_WIDTH-1:0] fifo_mem [0:FIFO_DEPTH-1];

    // Write and read pointers (need only log2(FIFO_DEPTH) bits)
    localparam PTR_WIDTH = $clog2(FIFO_DEPTH);
    reg [PTR_WIDTH-1:0] wr_ptr;
    reg [PTR_WIDTH-1:0] rd_ptr;

    // Counter to track number of elements in FIFO
    reg [PTR_WIDTH:0] fifo_count; // Can count up to FIFO_DEPTH

    // Write operation
    always @(posedge clk) begin
        if (rst) begin
            wr_ptr <= 0;
        end else if (wr_en && !full) begin
            fifo_mem[wr_ptr] <= din;
            wr_ptr <= wr_ptr + 1;
        end
    end

    // Read operation
    always @(posedge clk) begin
        if (rst) begin
            rd_ptr <= 0;
            dout   <= 0;
        end else if (rd_en && !empty) begin
```

```
            dout <= fifo_mem[rd_ptr];
            rd_ptr <= rd_ptr + 1;
        end
    end

    // FIFO count update
    always @(posedge clk) begin
        if (rst) begin
            fifo_count <= 0;
        end else begin
            case ({wr_en && !full, rd_en && !empty})
                2'b10: fifo_count <= fifo_count + 1; // Write only
                2'b01: fifo_count <= fifo_count - 1; // Read only
                2'b11: fifo_count <= fifo_count;     // Simultaneous read and write
                default: fifo_count <= fifo_count;   // No operation
            endcase
        end
    end

    // Status flags
    assign full         = (fifo_count == FIFO_DEPTH);
    assign empty        = (fifo_count == 0);
    assign almost_full  = (fifo_count >= FIFO_DEPTH - 1);
    assign almost_empty = (fifo_count <= 1);

endmodule
```